Diploma Project

REASONING ENFORCEMENT AND RIVISION IN ARGUMENTATION THEORIES AND BOOLEAN NETWORKS

Antriani Georgiou

University of Cyprus
Department of Computer Science

Department of Computer Science

May 2026

UNIVERSITY OF CYPRUS

Faculty of Pure and Applied Sciences

Department of Computer Science

REASONING ENFORCEMENT AND RIVISION IN ARGUMENTATION THEORIES AND BOOLEAN NETWORKS

Antriani Georgiou

Advisor:

Dr Dimopoulos Yannis

The Thesis was submitted in partial fulfillment of the requirements for obtaining the Computer Science degree of the Department of Computer Science of the University of Cyprus

May 2026

Acknowledgements

I would like to express my sincere gratitude to my advisor, Dr. Dimopoulos Yannis for his guidance, support and valuable feedback throughout the development of this diploma project.

I would also like to thank the Department of Computer Science at the University of Cyprus for providing me with the knowledge and resources needed to complete this work.

Finally, I would like to thank my family and friends for their continuous encouragement and support during my studies.

ABSTRACT

Boolean networks and abstract argumentation frameworks are two important formalisms used for representing complex systems through interactions between interconnected entities. Boolean networks are widely used in systems biology for modelling biological regulatory behaviour, while abstract argumentation frameworks are used for studying reasoning, conflicts and acceptability between arguments. Although these formalisms originate from different research areas, both involve notions of stable behaviour and consistency.

This thesis investigates the relationship between Boolean-network revision and formula-based argumentation enforcement under stable semantics.To achieve this a number of transformations between Boolean Networks and Argumentation Framework are established.

Several transformation pipelines and supporting programs were developed in order to convert Boolean-network models, SBML/SBML-qual biological models and binary CP-nets into Boolean or argumentation-based representations. In addition, programs were implemented for translating Boolean networks into APX frameworks, generating auxiliary constraints and analysing transformed models.

The experimental study was conducted using ARBoLoM, ModRev, bioLQM, BoolNet, CoQuiAAS and formula-based enforcement systems. The experiments compared repair behaviour, stable semantics, semantic-target optimisation and the effect of auxiliary arguments introduced during transformations.

The obtained results show that stable semantics provide a meaningful basis for comparison between Boolean-network revision and argumentation enforcement. However, the experiments also reveal important interpretability problems caused by auxiliary arguments generated during Boolean-to-APX transformations. To address this issue, a constraint-based enforcement strategy was proposed in order to restrict repairs on auxiliary structures and improve the correspondence between enforcement results and the original Boolean-network models.

Overall, the thesis demonstrates that argumentation-based enforcement can provide a useful complementary perspective for studying repair problems in Boolean regulatory systems while also highlighting important structural and semantic limitations of the transformations between the two representations.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Important abbreviations that have been used in the text and need explanation are briefly presented.

AF	Abstract Argumentation Framework
AI	Artificial Intelligence
AND-NOT	Restricted Boolean network representation using only AND and NOT operators
APX	Abstract Argumentation Framework format
ARBoLoM	ASP-based Revision of Boolean Logical Models
ASP	Answer Set Programming
BN	Boolean Network
BNet	Boolean Network format
BRN	Boolean Regulatory Network
CP-net	Conditional Preference Network
CPT	Conditional Preference Table
CoQuiAAS	Constraint-based Quick Abstract Argumentation Solver
HTML	HyperText Markup Language
ModRev	Model Revision Tool
RQ	Research Question
SBML	Systems Biology Markup Language
SBML-qual	Qualitative Systems Biology Markup Language
SAT	Boolean Satisfiability Problem

Declaration

I declare that this dissertation and any accompanying software are my own original work, conducted in full compliance with the University of Cyprus Code of Conduct for Research (https://www.ucy.ac.cy/legislation/kodikas-deontologias-kai-politikes/), and with full accountability on my part for the accuracy, integrity, and validity of all content.

1 Introduction

The modelling and analysis of complex reasoning systems is an important area in artificial intelligence and computational biology. Many real-world systems are composed of interacting elements whose behaviour depends on the state of other elements. Examples include biological regulatory systems, where genes and proteins influence each other, and logical reasoning systems, where arguments may support or attack each other. In such cases, computational models are used in order to represent the structure of the system, analyse its possible behaviours and study how the system can be modified when new information becomes available.

Boolean Networks are widely used for representing biological regulatory systems. In this formalism, each component of the system is represented as a Boolean variable that can take the value active or inactive. The behaviour of each variable is defined by a Boolean function that depends on other variables in the network. Although Boolean Networks are simpler than quantitative biological models, they are useful because they can capture the qualitative behaviour of a system even when detailed numerical data is not available. One important aspect of their analysis is the study of stable states and attractors, which describe possible long-term behaviours of the system.

However, constructing an accurate Boolean model is not always straightforward. As new experimental observations become available, an existing model may no longer agree with the observed behaviour. In such cases, the model may need to be revised or repaired. Boolean Network revision studies how a model can be modified so that it becomes consistent with a set of observations, while preserving as much of the original structure as possible. This is important because manual repair can be difficult, especially for larger networks where many possible changes may affect the final behaviour.

Another important formalism in artificial intelligence is Abstract Argumentation. An argumentation framework essentially represents arguments as nodes and attacks between arguments as directed edges. Different semantics are then used to determine which sets of arguments can be accepted. In particular, stable semantics is one of the central semantics used in argumentation. Argumentation enforcement studies how an argumentation framework can be modified so that a desired semantic condition becomes satisfied. This makes enforcement conceptually similar to revision, since both approaches involve modifying a system in order to satisfy specific requirements.

The main idea of this thesis is based on the connection between Boolean Network revision and argumentation enforcement. Stable states in Boolean Networks and stable extensions in argumentation frameworks are not identical concepts, but they both describe stable forms of be-

haviour under a given semantics. This creates the possibility of comparing the two formalisms and examining whether problems from Boolean Network revision can be represented and analysed through argumentation-based enforcement methods.

For this reason, the thesis investigates transformations between different model representations, including Boolean Network formats, APX argumentation frameworks and binary CP-net representations. Several tools are used in the experimental part of the work, such as ARBoLoM, ModRev, bioLQM, CoQuiAAS and formula-based enforcement systems. These tools allow the computation of stable states, attractors, stable extensions, repairs and enforcement solutions. Through these experiments, the thesis examines whether the results produced by different systems are equivalent, comparable or different.

An important part of the work concerns the transformation from Boolean Networks to Argumentation Frameworks. During this transformation, auxiliary variables may be introduced in order to represent Boolean functions and intermediate logical relations. This raises an important question: if an enforcement system changes an auxiliary variable, can this change be translated back into a meaningful repair of the original Boolean Network, or is it only an internal change of the encoding? This issue is central for understanding whether argumentation enforcement can be used not only to find acceptable solutions, but also to provide valid and meaningful repairs for Boolean models.

The thesis also considers CP-nets, which are preference-based models used to represent conditional preferences. Binary CP-net examples are transformed and analysed in order to examine how preference structures can be represented using Boolean-network-style encodings. This extends the work beyond biological models and argumentation frameworks, showing that similar transformation and analysis techniques can also be applied to preference reasoning.

Overall, the thesis shows that Boolean Networks, argumentation frameworks and CP-nets can be connected through model transformations and stable-semantics analysis. The experimental results indicate that the systems can sometimes produce comparable stable behaviours, especially when the translation preserves the relevant semantic information. At the same time, the results also show that care is needed when interpreting repairs, because changes at the level of the translated argumentation framework do not always correspond directly to meaningful changes in the original Boolean Network.

Therefore, the main outcome of this thesis is an experimental and methodological comparison of different reasoning systems under stable semantics. The work provides conversion programs, practical experiments and comparative results that help clarify the relationship between Boolean Network revision and argumentation enforcement. It shows where the two approaches can be connected, where they differ and what limitations must be considered when using argumentation-based methods for Boolean model repair.

In addition, the work provides a clearer understanding of the practical difficulties that appear when different reasoning systems are connected. Although Boolean Networks and argumentation frameworks can both be studied under stable semantics, their internal structures are different. Boolean Networks describe the evolution of variables through logical update functions, while argumentation frameworks describe acceptability through attacks between arguments. Because of this difference, a result that appears similar at the semantic level may still have a different structural interpretation.

This is why the thesis does not only compare final outputs, but also studies the way in which these outputs are produced. The experiments examine the original models, the transformed representations, the computed stable behaviours and the repairs or enforcement changes proposed by each system. This makes it possible to identify whether a solution is genuinely meaningful for the original model or whether it only satisfies the translated encoding. In particular, the distinction between original variables and auxiliary variables is important, because auxiliary variables are useful for encoding logical relations but do not always correspond to real components of the initial system.

Through this analysis, the thesis demonstrates that transformations between formalisms can be useful, but they must be interpreted carefully. When the transformation preserves the relevant stable behaviour, argumentation enforcement can provide a complementary way to analyse Boolean Network revision problems. However, when repairs affect only internal encoding elements, the result may not directly represent a valid repair of the original Boolean Network. Therefore, the work highlights both the potential and the limitations of using argumentation-based techniques for model revision.

The expected result of the thesis is not to prove that one system fully replaces the other, but to show how they can be compared in a systematic way. The implemented workflow allows models to be transformed, executed and evaluated across different tools. As a result, the thesis offers a practical basis for future work on connecting biological model revision, argumentation enforcement and preference-based reasoning through common semantic concepts.

2 Abstract Argumentation and Boolean Networks

2.0.1 Boolean Models

Boolean networks are mathematical and computational models used to represent the behaviour of dynamic systems through binary-valued variables and logical update rules. The formalism was originally introduced by Kauffman for the modelling of genetic regulatory systems and later became widely adopted in systems biology and network dynamics research.

Formally, a Boolean network is defined as a tuple:

$$BN = (V, F)$$

where:

$$V = \{x_1, x_2, \ldots, x_n\}$$

is a finite set of Boolean variables and

$$F = \{f_1, f_2, \ldots, f_n\}$$

is a set of Boolean update functions. Each variable:

$$x_i \in \{0, 1\}$$

can take one of two possible values, where typically:

- 0 represents an inactive or false state,
- 1 represents an active or true state.

Each update function:

$$f_i : \{0, 1\}^n \to \{0, 1\}$$

determines the next value of variable x_i according to the current values of a subset of variables in the network, known as its regulators.

A complete assignment of Boolean values to all variables defines a network state:

$$s = (x_1, x_2, \ldots, x_n)$$

Thus, a Boolean network with n variables contains:

$$2^n$$

possible states.

Boolean networks are particularly useful in systems biology because they provide a qualitative representation of regulatory behaviour without requiring precise quantitative kinetic parameters. Instead of modelling continuous concentrations or differential equations, Boolean models focus on activation and inhibition relations between components such as genes, proteins, or signalling molecules. This makes them especially suitable in biological domains where incomplete or uncertain quantitative information is available.

2.0.1.1 Boolean Update Schemes

The evolution of a Boolean network depends not only on the Boolean functions, but also on the update scheme used to apply them. Different update schemes may produce different dynamic behaviours and attractors.

2.0.1.1.1 Synchronous Update In the synchronous update scheme, all variables are updated simultaneously at each discrete time step. If the current state is:

$$s(t) = (x_1(t), x_2(t), \ldots, x_n(t))$$

then the next state is computed as:

$$s(t+1) = (f_1(s(t)), f_2(s(t)), \ldots, f_n(s(t)))$$

This approach is deterministic because every state has exactly one successor state.

2.0.1.1.2 Asynchronous Update In the asynchronous update scheme, only one variable is updated at a time. Different variables may be selected in different orders, producing multiple possible successor states from the same configuration.

Asynchronous updates are often considered biologically more realistic because biological processes usually do not evolve simultaneously at identical rates.

2.0.1.1.3 General and Probabilistic Updates

Additional update strategies also exist, including:

- fully asynchronous updates
- block-sequential updates
- stochastic or probabilistic updates
- priority-based update schemes

These approaches attempt to capture timing uncertainty and heterogeneous regulatory behaviour in complex biological systems.

2.0.1.2 Example of a Boolean Network

Listing 2.1: Simple Boolean network in BNet format

```
targets, factors
A, !B
B, A
```

The network above contains two Boolean variables:

$$V = \{A, B\}$$

with update functions:

$$f_A(B) = \neg B$$

and

$$f_B(A) = A$$

Variable A becomes active when B is inactive, while variable B follows the value of A. The interaction structure therefore defines a dependency relation between the two variables.

The possible states of this network are:

$$(0,0), (0,1), (1,0), (1,1)$$

Under synchronous updating, state transitions are generated by simultaneously applying both Boolean functions.

2.0.1.3 Stable States and Attractors

A stable state (or fixpoint) is a state that remains unchanged after applying the update functions. Formally, a state:

$$s^*$$

is a stable state if:

$$F(s^*) = s^*$$

where:

$$F = (f_1, f_2, \ldots, f_n)$$

represents the global transition function of the network.

Listing 2.2: Stable-state example

```
targets, factors
A, A
B, !A
```

For this network, one stable state is:

$$(A = 1, B = 0)$$

because:

$$f_A(1) = 1$$

and

$$f_B(1) = 0$$

Thus, after updating the variables, the network remains in the same configuration.

More generally, Boolean networks may contain attractors, which correspond to long-term dynamic behaviours of the system. Attractors may be:

- fixed-point attractors (stable states),

- cyclic attractors consisting of repeating sequences of states.

In systems biology, attractors are frequently interpreted as stable cellular behaviours, phenotypes, or biological equilibrium states.

In this thesis, stable states and attractors were validated using Boolean-network analysis tools such as BoolNet and bioLQM. These tools were used to compute fixpoints and compare the dynamic behaviour of Boolean models before and after transformations into argumentation-based representations. The preservation of stable semantic behaviour was an important requirement of the proposed transformations.

2.0.2 AND-NOT Boolean Networks

Another important representation used in the thesis is the AND-NOT Boolean-network form. In this restricted representation, Boolean update functions are expressed as conjunctions of literals, where each literal is either a variable or its negation. In other words, the logical structure of each rule is based on the use of AND and NOT operators. This type of representation is useful because it gives the network a more uniform logical form and can support the connection of Boolean models with other formalisms. In some cases, the transformation of general Boolean networks into AND-NOT form may require the introduction of additional or auxiliary variables in order to represent more complex Boolean expressions while preserving the relevant behaviour of the original model. As part of the methodology, programs were developed to read BNet models, parse their Boolean expressions and generate equivalent or related AND-NOT Boolean-network representations [Veliz-Cuba et al., 2012; Trinh et al., 2025].

Listing 2.3: AND-NOT Boolean network example

```
targets, factors
A, B & !C
B, A
C, !B
```

In this example, variable A becomes active only when B is active and C is inactive. Variable B follows the value of A, while variable C becomes active when B is inactive. This example shows how AND and NOT operators can be used to express regulatory dependencies in a restricted Boolean-network form.

2.0.3 CP-Nets

The thesis also studies CP-nets as a related preference-based formalism. CP-nets are graphical models used to represent conditional preferences under a ceteris paribus interpretation, meaning

that a preference is expressed while all other things are considered equal. In a CP-net, each variable is associated with a conditional preference table and the preference over the value of one variable may depend on the values of its parent variables. This makes CP-nets useful for representing structured qualitative preferences in a compact and intuitive way. In this thesis, the focus is on binary CP-nets, where each variable takes two possible values [Boutilier et al., 2004].

Listing 2.4: Binary CP-net example

```
VAR a : 0,1
VAR b : 0,1
VAR c : 0,1
VAR d : 0,1
VAR e : 0,1

PREF a : c=0 : 1 > 0
PREF a : c=1 : 0 > 1

PREF b : a=0 : 1 > 0
PREF b : a=1 : 0 > 1

PREF c : b=0 : 1 > 0
PREF c : b=1 : 0 > 1

PREF d : e=0 : 1 > 0
PREF d : e=1 : 0 > 1

PREF e : d=0 : 1 > 0
PREF e : d=1 : 0 > 1
```

In this example, all variables are binary and can take the values 0 or 1. The preferences of some variables depend on the value of another variable. For example, the preferred value of variable a depends on the value of c, while the preferred value of b depends on the value of a. This shows how CP-nets represent conditional qualitative preferences between variables.

CP-nets model preferences through local conditional preference rules. Reasoning in CP-nets is often based on transitions between outcomes using the concept of improving flips and flipping sequences.

An improving flip is a change in the value of a single variable while all other variables remain unchanged. Such a change is allowed only if it follows the preference rule of that variable under

the current assignment of its parent variables.

More formally, suppose that a variable X has domain values:

$$x \succ \bar{x}$$

under a particular parent assignment. If an outcome contains the less preferred value $\bar{x}$, replacing it with the preferred value x constitutes an improving flip.

A flipping sequence is a sequence of outcomes:

$$o_1 \rightarrow o_2 \rightarrow \cdots \rightarrow o_k$$

where each consecutive outcome differs in exactly one variable and each change corresponds to an improving flip according to the CP-net preference rules.

Flipping sequences are fundamental in dominance testing. If there exists an improving flipping sequence from outcome o to outcome o', then outcome o' is considered preferable to o according to the preference structure of the CP-net.

Listing 2.5: Simple CP-net preference example

```
A : a > !a
a : b > !b
!a : !b > b
```

Consider the outcomes:

$$(!a, !b)$$

and

$$(a, b)$$

An improving flipping sequence is:

$$(!a, !b) \rightarrow (a, !b) \rightarrow (a, b)$$

The first transition improves variable A because value a is preferred over $!a$. The second transition improves variable B under the condition that $A = a$.

Flipping sequences are important because they provide a transition-based interpretation of preferences. They also establish a connection between CP-net reasoning and state-transition systems, which is relevant to the transformation of CP-nets into Boolean-network and argumentation-based representations examined in this thesis.

2.0.4 SBML-qual

The thesis also considers SBML and SBML-qual files. SBML is a standard XML-based format for exchanging biological models, while SBML-qual is an SBML Level 3 package used for qualitative and logical models, including Boolean and multi-valued networks. In SBML-qual, the syntax is not line-based like BNet. Instead, model components are described using XML elements and attributes, such as qualitative species, transitions, inputs, outputs and function terms. A qualitative species represents a model variable, while transitions describe how the value of a variable is updated according to logical conditions. Because SBML-qual models may contain variables with more than two possible levels, they cannot always be analysed directly as Boolean networks. Since the work focuses on Boolean representations, multi-valued SBML-qual models are processed through booleanization before being used in later stages of the workflow. This allows models written in a richer biological exchange format to be transformed into Boolean representations that can be analysed together with the other tools and formats used in the thesis [Hucka et al., 2003; Chaouiya et al., 2015; Naldi et al., 2018].

2.1 Abstract Argumentation

2.1.1 Abstract Argumentation Frameworks

Abstract argumentation frameworks constitute another major component of this thesis. An abstract argumentation framework is formally defined as a pair:

$$AF = (A, R)$$

where A is a finite set of arguments and $R \subseteq A \times A$ is a binary attack relation between arguments. If $(a, b) \in R$, then argument a attacks argument b. This means that the acceptance of argument a may conflict with, or prevent, the acceptance of argument b.

Given a set of arguments $S \subseteq A$, the set S is conflict-free if there are no two arguments $a, b \in S$ such that $(a, b) \in R$. In other words, a conflict-free set does not contain arguments that attack each other.

Argumentation semantics are used to determine which sets of arguments, called extensions, can

be jointly accepted. In this thesis, particular emphasis is placed on stable semantics. A set $S \subseteq A$ is a stable extension of an argumentation framework $AF = (A, R)$ if and only if:

$$S \text{ is conflict-free}$$

and

$$\forall a \in A \setminus S,\ \exists b \in S \text{ such that } (b, a) \in R.$$

This means that a stable extension is a conflict-free set of arguments that attacks every argument outside the set. Stable extensions are important in this thesis because they provide a strong notion of stable acceptability, which can be compared with stable states in Boolean networks.

Stable semantics were selected as the main comparison point of the thesis because they provide a strong notion of equilibrium in both Boolean networks and argumentation frameworks. Stable states in Boolean networks and stable extensions in argumentation frameworks both correspond to configurations that remain semantically consistent under the rules of the system. This makes stable semantics particularly suitable for experimental comparison between the two representations.

Argumentation frameworks are represented in this thesis mainly using APX format. In this format, arguments and attacks are written as facts, for example `arg(a).` to declare an argument and `att(a,b).` to declare that argument a attacks argument b.

Listing 2.6: Simple argumentation framework in APX format

```
arg(a).
arg(b).
arg(c).

att(a,b).
att(b,a).
att(b,c).
```

In this example, arguments a and b attack each other, while argument b also attacks argument c. Under stable semantics, a stable extension must be conflict-free and must attack every argument outside the extension.

One stable extension is:

$$\{a, c\}$$

This set is conflict-free because neither a nor c attacks the other. It also attacks every argument outside the set, since argument a attacks b.

Another stable extension is:

$$\{b\}$$

This set is also conflict-free, and it attacks every argument outside it, since b attacks both a and c. Therefore, both $\{a, c\}$ and $\{b\}$ are stable extensions of this argumentation framework.

Listing 2.7: Stable extension example

```
arg(x).
arg(y).

att(x,y).
```

For this framework, the set $\{x\}$ is a stable extension because it is conflict-free and attacks every argument outside the set. Specifically, x attacks y, which is the only argument not included in the extension.

2.2 Relationship Between Boolean Networks and Abstract Argumentation

2.2.1 Conceptual Similarities

Although Boolean networks and abstract argumentation frameworks originate from different research areas, they share several conceptual similarities. In Boolean networks, the behaviour of a variable depends on the values of other variables through Boolean update rules. In abstract argumentation, the acceptability of an argument depends on the attacks received from other arguments. In both cases, the behaviour of the system is determined by interactions between interconnected entities.

2.2.2 Stable States and Stable Extensions

Stable states in Boolean networks and stable extensions in abstract argumentation both represent equilibrium notions within their respective formalisms. In Boolean networks, stable states correspond to configurations that remain unchanged after applying the Boolean update functions. Similarly, in abstract argumentation, stable extensions correspond to conflict-free sets of

arguments that attack every argument outside the extension[Dung, 1995].

This conceptual similarity motivates the comparison between Boolean networks and argumentation frameworks adopted in this thesis, since both notions capture forms of semantic stability and self-consistency.

3 Transformations Between Argumentation and Boolean Networks

This chapter presents the transformations implemented between Boolean-network representations and abstract argumentation frameworks. The main focus is the conversion between BNet and APX representations, because this is the transformation that directly connects Boolean networks with argumentation frameworks. Another transformation considered is that of CP-nets into Bollean Networks.

The main objective of these transformations was to compare experimentally the computational behavior of various stable-state and stable-extension systems. Moreover, the generated models were validated experimentally through stable-state computation, attractor analysis and stable-extension computation.

3.1 Preparatory Transformations Toward BNet Representation

Before comparing Boolean networks with argumentation frameworks, several models first had to be transformed into Boolean-network form. These transformations were useful because BNet acted as the common intermediate representation in the experimental workflow.

3.1.1 Transformation of General to AND/NOT Boolean Network

This transformation follows the method described in the results section of Veliz-Cuba et al. [Veliz-Cuba et al., 2014]. In that work, the authors show that a general finite dynamical system can be represented as an AND-NOT network by introducing additional nodes when necessary, while preserving a correspondence between the steady states of the original system and the transformed AND-NOT representation. Therefore, the purpose of the transformation is not simply syntactic simplification, but the construction of a restricted Boolean-network form that remains suitable for steady-state analysis.

General Boolean networks were first transformed into AND-NOT. This is needed because AND-NOT networks provide a more restricted and uniform logical representation where Boolean update rules are expressed through conjunctions and negations, which corresponds to the language of Argumentation Frameworks. The transformation required parsing Boolean expressions from BNet files. Simple expressions could be translated directly, while more complex expressions

required the introduction of auxiliary variables.

Listing 3.1: Input BNet model before AND-NOT transformation

```
targets, factors
A, 1
B, !A
C, A | B
D, C & !B
E, (A & C) | !D
F, E & !B
```

Listing 3.2: Resulting AND-NOT representation

```
A, 1
B, !A
C, !var1
D, (C & !B)
E, !var2
F, (E & !B)
var1, (!A & !B)
var2, (!var3 & D)
var3, A & C
```

In this example, auxiliary variables such as `var1`, `var2` and `var3` were introduced in order to encode Boolean sub-expressions that could not be represented directly in restricted AND-NOT form. This became important later in the thesis because repairs involving auxiliary variables may not correspond directly to meaningful modifications of the original Boolean model.

The workflow was also extended to SBML and SBML-qual biological models. Since SBML-qual models may contain multi-valued variables, they cannot always be analysed directly as Boolean networks. For this reason, bioLQM was used to booleanize SBML-qual models and convert them into BNet format. After booleanization, the generated Boolean models could be reused in the rest of the workflow.

3.1.2 Preference Nets and Boolean Networks

Binary CP-nets were also transformed into Boolean-network representations. CP-nets are preference-based models where variables have conditional preference rules depending on the values of parent variables. Although CP-nets were not the central comparison point of the thesis, their transformation showed that preference-based structures can also be represented in Boolean form.

Listing 3.3: Input binary CP-net

```
VAR x1 : 0,1
VAR x2 : 0,1
VAR x3 : 0,1

PREF x1 : TRUE : 1 > 0

PREF x2 : x1=1 : 1 > 0
PREF x2 : x1=0 : 0 > 1

PREF x3 : x1=1, x2=1 : 1 > 0
PREF x3 : x1=0, x2=0 : 0 > 1
```

Listing 3.4: Generated BNet representation from the CP-net

```
x1, 1
x2, x1
x3, (x1 & x2)
```

The transformation also establishes a connection between CP-net flipping sequences and the dynamics of the corresponding Boolean network. In CP-nets, an improving flipping sequence represents a sequence of preference-improving transitions between outcomes obtained by changing one variable at a time according to the preference rules. Similarly, in Boolean networks, the dynamics of the system are represented through transitions between Boolean states under the update rules of the network. Therefore, flipping sequences in the CP-net correspond to transition paths in the state-transition dynamics of the generated Boolean-network representation.

3.2 Transformation from Boolean Networks to APX Frameworks

The main transformation of the thesis concerned the generation of abstract argumentation frameworks from Boolean-network representations. This transformation created the practical connection between Boolean-network analysis and abstract argumentation reasoning.

The BNet-to-APX transformation was not applied directly to arbitrary Boolean expressions. Instead, as described in Section 3.1, the Boolean network was first transformed into an AND/NOT representation. This intermediate step was necessary because the APX encoding relies on logical dependencies that can be represented through conjunctions and negations. Therefore, auxiliary

variables introduced during the AND/NOT transformation were later translated into auxiliary arguments in the generated APX framework.

The transformation was implemented by developing a python program. In the produced APX framework, Boolean variables were represented as arguments, while logical dependencies between variables were represented through attack relations. The generated APX framework could then be analysed using CoQuiAAS in order to compute stable extensions.

The following example shows a Boolean-network model together with the generated APX framework.

Listing 3.5: Input BNet model for APX transformation

```
targets, factors
A, 1
B, !A
C, A | B
```

Listing 3.6: Generated APX framework

```
arg(a1).
arg(a2).
arg(a3).
arg(a4).
arg(false).
in(false).

att(a1, a2).
att(false, a2).
att(a4, a3).
att(a2, a4).
att(a1, a4).
att(false, a4).
```

In this transformation, Boolean variables became arguments in the APX framework, while attack relations encoded the logical dependencies of the original Boolean model. The argument `false` was introduced as a technical helper argument and `a4` was introduced as an auxiliary argument during the encoding of the Boolean expression.

The presence of auxiliary arguments shows that the transformation is not always a direct one-to-one mapping between Boolean variables and arguments. Some arguments are introduced only for representing intermediate logical expressions. This is important because such arguments are

part of the encoding and not part of the original Boolean model.

The generated APX frameworks were later analysed under stable semantics using CoQuiAAS. The obtained stable extensions were then compared with stable states computed from the original Boolean-network models. Through this process, the thesis examined whether the transformation preserved the stable behaviour of the original Boolean representation.

3.3 Transformation from APX Frameworks to Boolean Networks

The thesis also investigated the reverse transformation, from APX argumentation frameworks into Boolean-network representations. This transformation was implemented by a python program.

In this transformation, each argument becomes a Boolean variable and attack relations become inhibitory Boolean rules. Therefore, an argument can become active only when its attackers are inactive. This gives a direct way to represent argumentation attacks using Boolean update conditions.

The following example shows an APX framework together with the generated BNet representation.

Listing 3.7: Input APX framework

```
arg(a).
arg(b).
arg(c).

att(a,b).
att(b,c).
```

Listing 3.8: Generated BNet representation

```
targets, factors
a, 1
b, !a
c, !b
```

In this example, argument `a` has no attackers and therefore becomes permanently active in the Boolean-network representation. Argument `b` is attacked by `a`, so its Boolean update rule becomes `!a`. Similarly, argument `c` becomes active only when `b` is inactive.

This transformation allowed stable extensions computed through CoQuiAAS to be compared experimentally with stable states computed through Boolean-network tools. It is easy to show that the stable extensions of the initial argumentation theory coincide with the attractors of the corresponding Boolean Network.

4 Revision and Enforcement

This chapter presents the comparison between Boolean-network revision and formula-based argumentation enforcement under stable semantics. The chapter focuses on the repair behaviour of the examined systems, the effect of different observation modes and semantic targets and the interpretation problems introduced by auxiliary arguments during transformations between Boolean networks and APX frameworks.

The purpose of the experiments was not only to evaluate the individual systems independently, but also to investigate whether meaningful correspondences can be identified between Boolean-network revision and argumentation-based enforcement. Particular emphasis was placed on the understanding of repairs, the role of stable semantics and the structural differences between the two approaches.

Stable semantics were selected as the main comparison criterion because they provide the clearest notion of equilibrium behaviour in both Boolean networks and argumentation frameworks. Stable states correspond to fixed Boolean behaviours, while stable extensions correspond to conflict-free acceptable sets of arguments. This common equilibrium interpretation made stable semantics the most appropriate basis for comparing the two systems experimentally.

4.1 Revision and Enforcement Concepts

4.1.1 Model Revision

Model revision is an important concept in this thesis. In the context of Boolean regulatory networks, model revision refers to the process of modifying an existing Boolean model so that it becomes consistent with a set of new observations or experimental constraints. These observations may describe expected stable states, time-series behaviours or other dynamical properties. A revision system first checks whether the original model satisfies the observations. If the model is inconsistent, the system searches for repairs that restore consistency while preserving as much of the original model as possible [Gouveia et al., 2020].

Listing 4.1: Stable-state observation example

```
experiment(1).
observation(1,A,1).
observation(1,B,0).
```

This observation requires variable A to be active and variable B to be inactive in a stable state of

the network. If the original Boolean model does not satisfy this observation, a revision system searches for repairs that restore consistency.

4.1.2 Formula-Based Enforcement

Formula-based enforcement is the argumentation-side counterpart studied in this thesis. In abstract argumentation, enforcement refers to the modification of an argumentation framework so that a desired set of arguments obtains a required acceptability status under a selected semantics. Formula-based enforcement extends this idea by allowing semantic and syntactic requirements to be expressed through propositional formulas. Some constraints may be hard, meaning that they must be satisfied, while others may be soft, meaning that violations are allowed but contribute to the optimisation cost [Baumann and Brewka, 2010; Coste-Marquis et al., 2015; Doutre and Mailly, 2018].

Listing 4.2: Formula-based enforcement query example

```
enf(a).
enf(b).
```

This query requires arguments a and b to become accepted under the selected semantics. The enforcement system searches for modifications of the original framework that satisfy these requirements while minimising changes.

4.2 Boolean-Network Revision and Argumentation Enforcement

The first group of experiments compared Boolean-network revision with formula-based argumentation enforcement under stable semantics. The objective was to examine whether similar repair behaviour can be observed when both systems are required to satisfy analogous stable conditions.

Although the two systems operate on fundamentally different structures, they both attempt to restore consistency through minimal modifications. In Boolean-network revision, repairs are expressed as changes to regulators, regulator signs or Boolean functions. In formula-based enforcement, repairs are expressed as modifications of attack relations between arguments.

The experiments were conducted using ARBoLoM and ModRev for Boolean-network revision and formula-based enforcement under stable semantics for the argumentation side. The experiments focused mainly on stable-state observations because the enforcement framework does not

directly support time-series biological constraints in the same way as Boolean-network revision systems.

4.2.1 Example 1: Consistent Stable-State Behaviour

In the first experiment, an ARBoLoM Boolean regulatory model containing three variables was tested against stable-state observations. The observations required all variables to have value 1 simultaneously. ARBoLoM reported that the model was already consistent with the observations and therefore no repair was required.

A corresponding argumentation framework was then constructed using equivalent variable names as arguments. Since no attack relations existed between the arguments, all arguments could be accepted simultaneously under stable semantics. The formula-based enforcement solver also returned an optimal solution with cost 0.

Table 4.1: Result of the first stable-state comparison example

System	**Input**	**Requested condition**	**Result**
ARBoLoM	Boolean network	Stable-state observations satisfied	Consistent model, no repair required
Formula enforcement	APX framework	Arguments accepted under stable semantics	Optimal solution found with cost 0

This experiment demonstrates that the two approaches may exhibit semantically equivalent behaviour under stable semantics, even though they operate on different underlying structures.

Listing 4.3: Representative formula-enforcement result

```
s OPTIMUM FOUND
o 0
arg(v1).
arg(v2).
arg(v3).
```

The experiment also illustrates that simple stable-state behaviour can often be represented consistently in both formalisms. In this case, no repairs were necessary because the required stable configuration was already supported by the original structures.

4.2.2 Example 2: Repair Through Structural Modifications

The second experiment used a corrupted Boolean regulatory model executed in ARBoLoM. The model contained four Boolean variables and the observations required all variables to stabilise to value 0. ARBoLoM detected that the model was inconsistent with the observations and proposed a repair. The repair modified the internal Boolean functions and regulators structure.

A corresponding formula-based enforcement benchmark was executed under stable semantics. The argumentation framework contained twenty arguments and a set of attack relations. The solver successfully computed an optimal enforcement solution by modifying attacks in the framework until the target conditions became satisfied.

Table 4.2: Comparison of repair behaviour in the second example

System	**Model size**	**Repair type**	**Output**
ARBoLoM	small	Repair on Boolean functions and regulators	Consistent model obtained after repair
Formula enforcement	small	Modification of attack relations	Optimal enforcement solution found

The obtained results indicate that the two systems behave similarly at a high conceptual level because both attempt to restore stable behaviour through minimal structural modifications. However, the meaning of the repairs differs significantly. Boolean-network revision produces repairs directly on regulatory functions and dependencies, whereas formula-based enforcement modifies only the argumentation structure.

Execution-time observations also showed differences between the two systems. Small Boolean-network repairs were generally computed efficiently by ARBoLoM, while enforcement experiments involving larger APX frameworks required more optimisation steps due to the larger search space of attack modifications.

4.2.3 Example 3: More Complex Repair Scenario

The third experiment involved a more complex corrupted Boolean regulatory model. ARBoLoM reported that the model was inconsistent with the stable-state observations and proposed repairs. These repairs involved regulator-sign changes, additions and removals of regulators and modifications of the Boolean functions.

The corresponding formula-based enforcement experiment was executed on a larger argumentation framework containing twenty arguments and multiple attack relations. The solver again

computed an optimal solution, but the optimisation cost increased and the enforcement process required more structural modifications.

Table 4.3: Summary of the third comparison example

System	Main result	Repair details	Interpretation
ARBoLoM	Inconsistent model repaired	Repair on Boolean functions and regulators	Boolean-level repair affecting regulatory functions and signs
Formula enforcement	Optimal solution found	Multiple attack-relation modifications	Argumentation-level repair affecting the APX structure

The experiments demonstrated that the complexity of the repair process increases significantly as the structure of the model becomes richer. The results also showed that semantic equivalence between the two systems does not necessarily imply structural equivalence between the produced repairs.

The third experiment also revealed scalability differences between the systems. Larger APX frameworks frequently generated larger optimisation costs and required additional enforcement iterations, while Boolean-network revision focused mainly on local regulatory modifications.

4.3 Experimental Modelling of Boolean-Network Repair through Argumentation Enforcement

The previous examples demonstrated conceptual similarities between Boolean-network revision and formula-based argumentation enforcement. However, these examples were intentionally small and were mainly used to illustrate the basic repair behaviour of the two approaches. In order to investigate the relationship more systematically, an automated experimental methodology was developed and evaluated on a larger collection of randomly generated theories.

The objective of the experiments was to examine whether repair problems originating from Boolean AND/NOT networks can be represented as enforcement problems on argumentation frameworks and whether the two systems exhibit comparable behaviour under stable semantics. The experiments were designed to provide a common semantic basis for comparison while preserving the individual characteristics of each formalism.

The experimental process begins with the automatic generation of argumentation frameworks of different sizes and densities. These frameworks are transformed into equivalent Boolean-

network representations and subsequently subjected to controlled structural modifications. The purpose of the corruption phase is to introduce inconsistencies that require repair, thereby creating instances suitable for both revision and enforcement.

After the corruption step, the resulting Boolean networks are transformed back into argumentation frameworks and analysed under stable semantics. The stable extensions of the corrupted frameworks are then computed and used as the common semantic representation for the remainder of the comparison.

For the enforcement approach, each stable extension is translated into a positive semantic target. Consequently, every stable extension becomes a desired acceptable set of arguments. Arguments that do not appear in any stable extension are translated into negative targets. The resulting enforcement specification therefore captures the stable behaviour of the corrupted framework and is used to guide the enforcement solver.

For the revision approach, the same stable extensions are transformed into stable-state observations. Each stable extension is interpreted as a desired stable configuration and is encoded as an observation set for the revision system. The revision solver then attempts to repair the original Boolean-network model so that the required stable states become consistent with the model.

The use of the same stable-semantic information for both approaches ensures that the comparison is performed under equivalent semantic requirements. Rather than comparing unrelated repair tasks, the methodology compares the behaviour of the two systems when attempting to satisfy the same stable conditions derived from the corrupted theory.

The entire process was fully automated through a single experimental pipeline. The pipeline generated the theories, performed all required transformations, constructed the semantic targets and observations, executed both repair systems and recorded execution statistics automatically. This automation allowed the evaluation of multiple theory sizes and densities while ensuring a consistent experimental methodology.

4.3.1 Experimental Results

The experimental evaluation was conducted on randomly generated theories of different sizes and densities. For each instance, the complete pipeline was executed automatically. The execution times of both revision and enforcement were recorded together with the number of stable extensions identified in the corrupted framework.

Table 4.4: Execution times of revision and enforcement under stable semantics

Size	Density	Stable Extensions	Enforcement Time (s)	Revision Time (s)
10	0.6	1	3.51	0.65
10	0.8	2	3.06	0.69
15	0.2	1	2.98	0.70
15	0.4	1	3.32	0.65
15	0.6	2	4.89	0.83
15	0.8	3	3.46	0.70
20	0.2	2	4.82	0.67
20	0.4	1	3.08	0.68
20	0.6	2	4.88	0.68
20	0.8	1	3.02	0.73
25	0.2	1	2.93	0.63
25	0.4	1	2.96	0.67
25	0.6	1	2.92	0.72
25	0.8	2	3.09	0.88
30	0.2	2	3.09	0.63
30	0.4	3	2.99	0.74
30	0.6	2	3.11	0.78
30	0.8	1	3.40	0.82

The obtained results indicate that both approaches were capable of successfully solving the generated repair problems whenever stable extensions existed for the corrupted framework. Instances that did not admit stable extensions could not be transformed into meaningful semantic targets or stable-state observations and were therefore excluded from the comparative evaluation.

A consistent observation across all successful executions was the difference in execution time between the two approaches. The revision system produced repairs in less than one second for all evaluated instances, whereas formula-based enforcement generally required between three and five seconds. This difference can be attributed to the optimisation process performed by the enforcement solver, which must search for minimal structural modifications satisfying the generated semantic targets.

Despite these computational differences, both approaches successfully produced repaired solutions under equivalent stable-semantic requirements. The experiments therefore provide evidence that Boolean-network repair can be modelled experimentally as an argumentation-enforcement problem and that meaningful comparisons between the two paradigms can be performed through

stable semantics.

Furthermore, the experiments demonstrated that stable extensions provide an effective semantic bridge between Boolean-network revision and argumentation enforcement. By translating stable extensions into enforcement targets and revision observations, it becomes possible to compare the behaviour of the two systems using a common semantic interpretation while preserving the underlying characteristics of each formalism.

Overall, the results support the central hypothesis of this thesis: although Boolean-network revision and argumentation enforcement operate on different structures and employ different repair mechanisms, stable semantics provide a sufficiently strong correspondence to enable meaningful experimental comparisons between the two approaches.

4.4 Auxiliary Arguments and Repair Interpretation

One of the most important findings of the thesis concerns the role of auxiliary arguments introduced during the transformation from Boolean networks into APX frameworks.

During the BNet-to-APX transformation, additional arguments may be introduced in order to encode internal Boolean expressions. These auxiliary arguments do not correspond directly to original Boolean variables, but instead represent intermediate logical structures created during the encoding process.

The following small BNet model was used:

Listing 4.4: Input BNet model

```
targets, factors
A, 1
B, !A
C, A | B
```

After transformation into APX format, the generated framework contained one auxiliary argument:

Listing 4.5: Generated APX framework

```
arg(a1).
arg(a2).
arg(a3).
arg(a4).
arg(false).
in(false).
```

```
att(false,a2).
att(a1,a2).
att(a4,a3).
att(false,a4).
att(a2,a4).
att(a1,a4).
```

Table 4.5: Mapping between BNet variables and APX arguments

BNet element	**APX argument**	**Type**
`A`	`a1`	Original Boolean variable
`B`	`a2`	Original Boolean variable
`C`	`a3`	Original Boolean variable
Auxiliary Boolean expression	`a4`	Auxiliary argument introduced during encoding
Technical false node	`false`	Technical helper argument

The enforcement query required the auxiliary argument `a4` to become accepted:

Listing 4.6: Enforcement query on auxiliary argument

```
target(a4).
```

The solver returned an optimal solution with cost 2. During the repair process, attacks involving the auxiliary argument were modified.

Originally, the APX framework contained:

Listing 4.7: Original attacks involving the auxiliary argument

```
att(a1,a4).
att(a2,a4).
att(false,a4).
```

After enforcement, the framework became:

Listing 4.8: Enforced attacks involving the auxiliary argument

```
att(a2,a4).
att(false,a4).
att(a4,false).
```

Thus, the repair removed the attack `att(a1,a4)` and introduced the attack `att(a4,false)`.

Table 4.6: Interpretation of repairs involving auxiliary arguments

APX repair	**Affected element**	**Interpretation in original BNet**
Remove `att(a1,a4)`	Auxiliary argument	Modification of encoding structure rather than original Boolean dependency
Add `att(a4,false)`	Auxiliary argument	Artificial repair affecting internal APX representation

This experiment revealed an important limitation of direct APX-based enforcement on transformed Boolean models. The enforcement solver operates purely on the argumentation structure and therefore cannot distinguish automatically between original Boolean variables and helper structures introduced during encoding.

Consequently, repairs involving auxiliary arguments may satisfy the enforcement objective while remaining difficult to interpret biologically or logically at the Boolean-network level.

The experiment therefore suggests that semantic equivalence between Boolean networks and argumentation frameworks does not necessarily imply repair equivalence. Even when stable semantics remain comparable between the two representations, repairs performed on auxiliary encoding structures may fail to correspond to meaningful modifications of the original Boolean regulatory model.

4.5 Constraint-Based Direction for Future Work

A possible direction for addressing the auxiliary-repair problem is to restrict the enforcement process using hard constraints on attack relations.

The proposed solution is to classify the generated APX arguments into two categories:

- original arguments, corresponding directly to Boolean variables,
- auxiliary arguments, introduced only for encoding Boolean expressions.

After this classification, attacks involving auxiliary arguments could remain fixed during enforcement, while the solver would only be allowed to modify attacks involving original arguments.

Conceptually, this would turn the enforcement process into a more structure-preserving repair method. The solver would still search for optimal solutions under stable semantics, but it would not be allowed to satisfy the enforcement query through modifications of encoding artefacts.

Table 4.7: Proposed constraint-based repair strategy

Attack type	**Repair allowed**	**Reason**
Original-to-original	Yes	The attack corresponds directly to original Boolean variables
Original-to-auxiliary	No	The attack belongs to the encoding structure
Auxiliary-to-auxiliary	No	The attack connects helper structures introduced during transformation
Auxiliary-to-original	No	The repair may not correspond to meaningful Boolean-network modifications

This approach could improve the interpretability of repairs and make the comparison between Boolean-network revision and argumentation enforcement more meaningful.

4.6 Limitations

Although the experiments demonstrated meaningful correspondences between Boolean-network revision and argumentation enforcement, several limitations remain.

First, the relationship between the two formalisms is not exact. Boolean networks model regulatory dynamics explicitly through update functions, whereas argumentation frameworks model acceptability through attack relations. As a result, some transformations preserve stable behaviour only approximately.

Second, the transformations frequently introduce auxiliary variables and helper arguments. These structures are necessary for encoding complex Boolean expressions, but they complicate the interpretation of repairs produced after enforcement.

Another limitation concerns the supported semantics. The enforcement experiments focused mainly on stable semantics because they provided the clearest comparison with stable states in

Boolean networks. Other semantics such as admissible or preferred semantics were not examined extensively in this thesis.

Finally, the experiments were conducted mainly on small and medium-sized examples. Larger biological models may introduce additional scalability and optimisation challenges that require further investigation.

5 System Design for the Auxiliary Variables Problem and Experimental Results

This chapter presents the implementation developed during this thesis and the experimental evaluation performed using Boolean-network revision tools and formula-based argumentation enforcement systems.

The chapter first describes the overall transformation and enforcement workflow that was implemented to enable interoperability between Boolean networks and argumentation frameworks. It then presents the developed programs, the auxiliary-constraint generation mechanism and the experiments conducted under stable semantics. Finally, the obtained results are analysed and compared.

The implementation was designed with the goal of creating a reproducible workflow capable of transforming models between different representations and evaluating them systematically using both Boolean-network and argumentation-based tools. Particular emphasis was placed on stable-state preservation, repair interpretability and the handling of auxiliary variables introduced during transformations.

5.1 System Overview

One of the main objectives of this thesis was to construct a workflow capable of transforming Boolean-network models into argumentation frameworks and then applying formula-based enforcement techniques under stable semantics.

The implemented system combines Boolean-network processing, Boolean transformation, APX generation, auxiliary-argument handling and formula-based enforcement into a unified pipeline. The workflow was designed to support direct experimentation between Boolean-network revision systems and argumentation-based repair approaches.

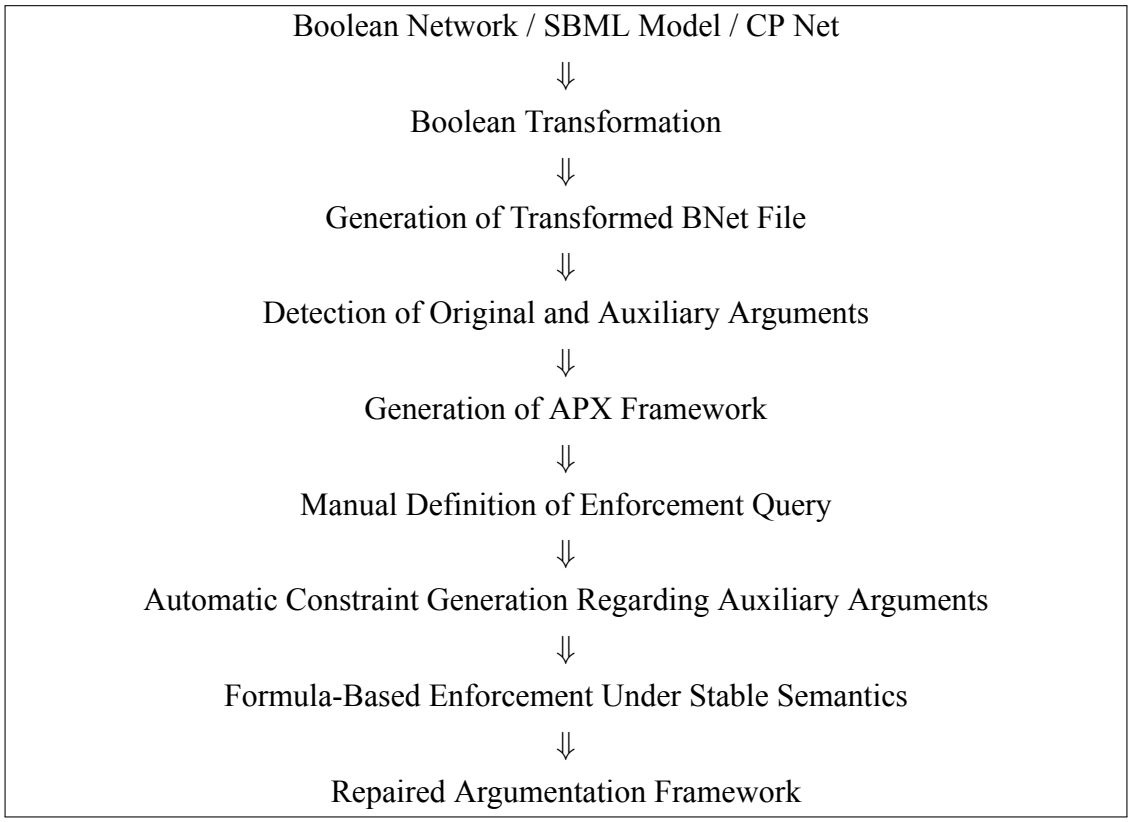

Figure 5.1: Overall workflow implemented in the thesis

The implemented workflow supports both unrestricted enforcement and constrained enforcement. In unrestricted enforcement, all attacks in the APX framework may be modified. In constrained enforcement, attacks involving auxiliary arguments are restricted in order to preserve the Boolean encoding structure introduced during the translation process.

The workflow was designed in a modular way so that each transformation stage could be executed independently. This made it easier to validate intermediate representations, identify errors in the generated files, and compare the outputs of different tools separately.

5.2 Translation and Transformation Pipeline

A major part of the implementation focused on the transformation of Boolean-network models into APX frameworks suitable for argumentation enforcement.

The program `translate.py` takes the original `.bnet` file as input and produces a transformed Boolean-network file, for example `input_out.bnet`. During this step, it also generates the file `original_args.txt`, which stores the original Boolean variables before the introduction of auxiliary variables.

The program `attractors.py` is then applied to the transformed Boolean network and generates

the corresponding APX framework, for example `input_out_out.apx`. The enforcement query file, such as `input_out_out.apx.query`, is not generated automatically by the pipeline. It is created manually, depending on the desired semantic target or enforcement requirement of each experiment.

5.2.1 Boolean Expression Transformation

The translation procedure performs several transformations on the input Boolean expressions:

- insertion of explicit parentheses,
- elimination of double negations,
- conversion of disjunctions into conjunction and negation form using De Morgan transformations,
- introduction of auxiliary variables for intermediate expressions,
- generation of the transformed BNet output,
- generation of the list of original Boolean variables.

The following example illustrates the transformation process:

Listing 5.1: Example Boolean-network input

```
targets, factors
A, 1
B, !A
C, A | B
```

After transformation, the resulting Boolean network contains an additional helper variable:

Listing 5.2: Transformed Boolean-network output

```
targets, factors
A, 1
B, !A
C, !var1
var1, (!A & !B)
```

In this example, the auxiliary variable `var1` is introduced in order to encode the original disjunction `A | B`. Using De Morgan's law, the expression is rewritten as the negation of a conjunction of negated variables. This makes the Boolean expression compatible with the later APX encoding.

The experiments showed that the number of generated auxiliary variables increases with the complexity of the Boolean expressions. Models containing several disjunctions produced larger transformed representations and more intermediate encoding structures.

5.2.2 Generation of Original Arguments

During translation, the program also extracts and stores the original Boolean variables separately. This information is later used to distinguish original arguments from auxiliary arguments during constrained enforcement.

The generated file `original_args.txt` contains the list of original Boolean variables:

Listing 5.3: Example original arguments file

```
a1
a2
a3
```

This separation is important because auxiliary variables do not correspond directly to original variables of the Boolean network. They are introduced only as technical elements of the encoding. Therefore, allowing the enforcement solver to freely modify them may lead to repairs that are valid in the APX framework but difficult to interpret as meaningful changes in the original Boolean-network model.

The identification of original arguments later became one of the key components of the constraint-based enforcement approach developed in this thesis.

5.3 APX Generation and Manual Query Definition

After the Boolean network is transformed into the required form, the program `attractors.py` is used to generate the corresponding APX framework.

The generated APX file contains arguments and attacks that encode the transformed Boolean-network structure. The arguments corresponding to the original Boolean variables are later identified using the file `original_args.txt`. All additional arguments are treated as auxiliary or technical arguments.

A typical APX output contains facts of the following form:

Listing 5.4: Example APX framework

```
arg(a1).
arg(a2).
```

```
arg(a3).
arg(var1).

att(a1,a2).
att(var1,a3).
```

The query file is not produced automatically. It is manually created for each experiment according to the desired enforcement target. For example, a query may require that a specific argument is credulously accepted under stable semantics.

Listing 5.5: Example manually defined query file

```
pos(a1).
```

This manual step allows each experiment to define the semantic requirement that the enforcement solver must satisfy. As a result, the same APX framework can be tested under different enforcement targets by changing only the query file.

5.4 Auxiliary Constraint Generation

A second implementation component focused on preventing repairs on auxiliary APX structures.

The developed program `generate_aux_constraints.py` automatically generates constraints that restrict attacks involving auxiliary arguments. The purpose of this step is to preserve the technical encoding introduced during the Boolean-to-APX transformation.

The script reads:

- the generated APX framework,
- the list of original arguments,
- the existing attack relations.

It then automatically classifies arguments into:

- original arguments,
- auxiliary or technical arguments.

5.4.1 Constraint Generation Strategy

The implemented strategy identifies every attack relation where at least one endpoint corresponds to an auxiliary argument.

If an attack already exists in the APX framework, the generated constraint file can preserve it using:

```
att(x,y).
```

If an attack does not exist and contains an auxiliary argument, the generated file can prevent its addition using:

```
-att(x,y).
```

The generated constraints are written into a separate constraint file and are later passed directly to the enforcement solver.

This mechanism was introduced because unrestricted enforcement may produce repairs on encoding structures instead of repairs on the original Boolean-network dependencies. Such repairs can reduce the enforcement cost, but they may not correspond to meaningful changes in the original model.

6 Results and Conclusions

This thesis investigated the relationship between Boolean-network revision and formula-based argumentation enforcement under stable semantics. The objective was not only to compare two different reasoning paradigms, but also to examine whether repair problems originating from Boolean regulatory networks can be represented and analysed through argumentation-based enforcement techniques.

To support this investigation, a complete transformation workflow was developed connecting Boolean networks, AND-NOT Boolean networks and abstract argumentation frameworks. Additional transformations involving SBML/SBML-qual models and binary CP-nets were also implemented in order to provide a broader evaluation of the proposed methodology. These transformations enabled models originating from different domains to be analysed under a common stable-semantics perspective.

The experimental evaluation demonstrated that stable semantics provide a meaningful basis for comparison between the two formalisms. Stable states computed in Boolean-network representations and stable extensions computed in argumentation frameworks frequently exhibited comparable semantic behaviour. This confirms that stable semantics can serve as a common conceptual bridge between the two domains despite their fundamentally different modelling assumptions.

At the same time, the experiments revealed important structural differences between revision and enforcement. Boolean-network revision modifies regulatory functions, regulators and regulatory signs in order to restore consistency with observations. In contrast, formula-based enforcement modifies attack relations in an argumentation framework in order to satisfy semantic targets. Consequently, although both approaches attempt to restore desired behaviour through minimal modifications, the resulting repairs operate at different representational levels and do not necessarily correspond directly to one another.

A central contribution of this work concerns the study of auxiliary variables introduced during Boolean-to-APX transformations. The experiments showed that enforcement solutions may exploit auxiliary encoding structures in order to satisfy enforcement objectives. While such repairs are valid from the perspective of the argumentation framework, they do not always correspond to meaningful modifications of the original Boolean model. This observation highlights an important challenge when transferring repair operations across different representations. To address this issue, a constraint-based methodology was developed for distinguishing original model components from auxiliary encoding structures.

The comparative experiments further demonstrated that both revision and enforcement can suc-

cessfully solve repair problems derived from equivalent stable-semantic requirements. However, the computational behaviour of the two approaches differs. Revision systems generally produced solutions more efficiently, whereas formula-based enforcement required additional optimisation effort due to the search for minimal attack modifications satisfying semantic constraints. Despite these differences, both approaches were capable of producing valid repaired solutions under the same semantic objectives.

The results therefore support the main hypothesis of this thesis: stable semantics provide a sufficiently strong correspondence to enable meaningful experimental comparisons between Boolean-network revision and argumentation enforcement. While the two paradigms are not equivalent and operate on different structures, they can nevertheless be related through carefully designed transformations and common semantic interpretations.

Overall, this thesis contributes a transformation-based methodology for studying interoperability between Boolean networks and abstract argumentation frameworks, an experimental comparison between revision and enforcement under stable semantics and a practical strategy for improving the interpretability of repairs in transformed models. The work demonstrates both the opportunities and the limitations of using argumentation-based techniques to analyse and repair Boolean regulatory systems.

Several directions for future work remain open. Larger benchmark collections and biological regulatory models could be investigated in order to evaluate scalability more extensively. Additional argumentation semantics beyond stable semantics could also be considered. Furthermore, more sophisticated constraint-generation mechanisms may improve the correspondence between enforcement modifications and biologically meaningful Boolean-network repairs.

In conclusion, this thesis shows that stable-semantics analysis provides a useful and principled framework for connecting Boolean-network revision and argumentation enforcement. Although the two approaches remain distinct in terms of representation and repair mechanisms, the performed experiments demonstrate that they can be compared systematically through semantic-preserving transformations. The obtained results establish a foundation for future research on integrating model revision, argumentation reasoning and preference-based systems within a unified semantic framework.

BIBLIOGRAPHY

[1] Albert, R., Wang, R. S., and Barabási, A.-L. Boolean modeling of cellular signaling networks. *Methods in Enzymology*, 467:133–160, 2008.

[2] Aleixo, F., Gouveia, F., Lynce, I., and Monteiro, P. T. Revising Boolean logical models of biological regulatory networks. In *Proceedings of KR 2023*, pages 16–26, 2023.

[3] Baumann, R. and Brewka, G. Expanding argumentation frameworks: Enforcing and monotonicity results. In *Proceedings of COMMA 2010*, pages 75–86. IOS Press, 2010.

[4] Baumann, R., Doutre, S., Heras, S., Levy, N., Mailly, J.-G., and Woltran, S. Enforcement in formal argumentation. *IfCoLog Journal of Logics and their Applications*, 8(3):567–610, 2021.

[5] Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H. H., and Poole, D. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

[6] Chaouiya, C., Naldi, A., and Thieffry, D. Logical modelling of gene regulatory networks with GINsim. *Methods in Molecular Biology*, 1021:463–479, 2013.

[7] Coste-Marquis, S., Konieczny, S., Mailly, J.-G., and Marquis, P. Extension enforcement in abstract argumentation as an optimization problem. In *Proceedings of IJCAI 2015*, pages 2876–2882, 2015.

[8] Doutre, S. and Mailly, J.-G. Constraints and changes: A survey of abstract argumentation dynamics. *Argument & Computation*, 9(3):223–248, 2018.

[9] Dung, P. M. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.

[10] Gouveia, F., Lynce, I., and Monteiro, P. T. ModRev: Model revision tool for Boolean logical models of biological regulatory networks. In *Computational Methods in Systems Biology*, pages 339–348. Springer, 2020.

[11] Hucka, M. et al. The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[12] Ignatiev, A., Morgado, A., and Marques-Silva, J. RC2: an efficient MaxSAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 11:53–64, 2019.

[13] Kauffman, S. A. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467, 1969.

[14] Lagniez, J.-M., Lonca, E., and Mailly, J.-G. CoQuiAAS: A constraint-based quick abstract argumentation solver. In *Proceedings of ICTAI 2015*, pages 582–588, 2015.

[15] Müssel, C., Hopfensitz, M., and Kestler, H. A. BoolNet: An R package for generation, reconstruction and analysis of Boolean networks. *Bioinformatics*, 26(10):1378–1380, 2010.

[16] Naldi, A. BioLQM: A Java toolkit for the manipulation and conversion of logical qualitative models of biological regulatory networks. *Frontiers in Physiology*, 9:1605, 2018.

[17] Ribeiro, A. S., Kauffman, S. A., Lloyd-Price, J., Samuelsson, B., and Socolar, J. E. S. Mutual information in random Boolean models of regulatory networks. *Physical Review E*, 77(1), 2008.

[18] Trinh, M., Veliz-Cuba, A., and Laubenbacher, R. AND-NOT representations of Boolean network models and their applications. *Journal of Computational Biology*, 32(1):44–59, 2025.

[19] Veliz-Cuba, A., Aguilar, B., Hinkelmann, F., and Laubenbacher, R. Steady state analysis of Boolean molecular network models via model reduction and computational algebra. *BMC Bioinformatics*, 13:221, 2012.

[20] Wallner, J. P., Niskanen, A., and Järvisalo, M. Complexity results and algorithms for extension enforcement in abstract argumentation. *Journal of Artificial Intelligence Research*, 60:1–40, 2017.

[21] CoQuiAAS Documentation. Constraint-based Quick Abstract Argumentation Solver. https://www.cril.univ-artois.fr/coquiaas/

[22] ICCMA. International Competition on Computational Models of Argumentation. https://iccma2023.github.io/

[23] ARBoLoM Project Repository. https://github.com/fpaleixo/arbolom

[24] bioLQM Documentation. Logical qualitative model manipulation toolkit. https://colomoto.github.io/biolqm/

[25] BoolNet Documentation. Boolean network analysis package for R. https://cran.r-project.org/package=BoolNet

[26] SBML-qual specification. Qualitative models package for SBML Level 3. http://sbml.org/Documents/Specifications/SBML_Level_3/Packages/qual